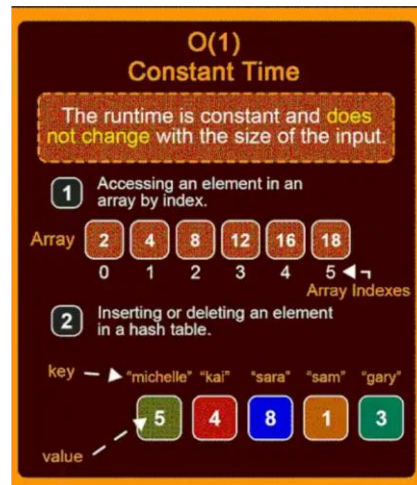


Big O Notation The Secret to Writing Efficient Algorithms:

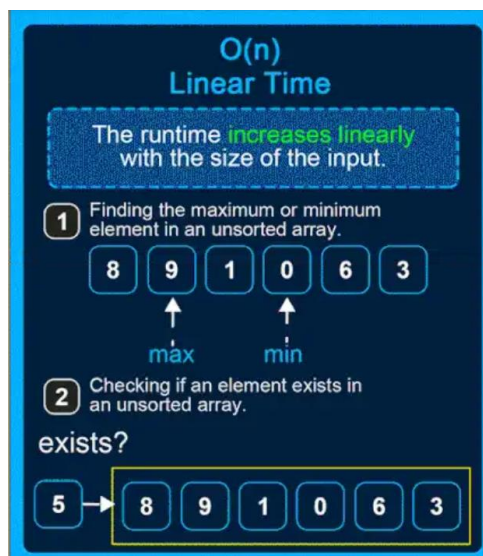
1. $O(1)$

This is the constant time notation. The runtime remains steady regardless of input size. For example, accessing an element in an array by index and inserting/deleting an element in a hash table.



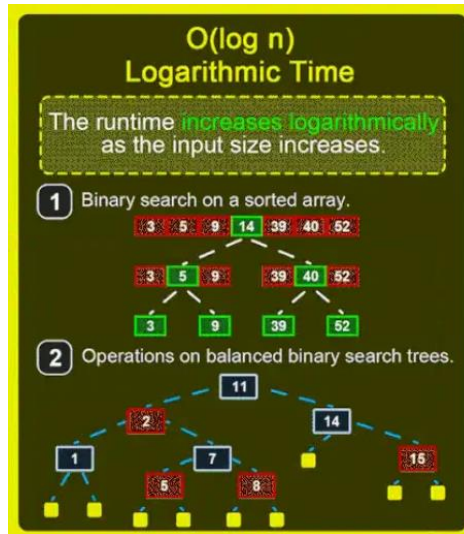
2. $O(n)$

Linear time notation. The runtime grows in direct proportion to the input size. For example, finding the max or min element in an unsorted array.



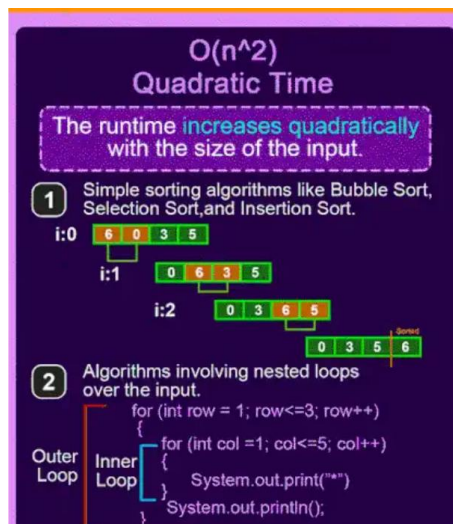
3. $O(\log n)$

Logarithmic time notation. The runtime increases slowly as the input grows. For example, a binary search on a sorted array and operations on balanced binary search trees.



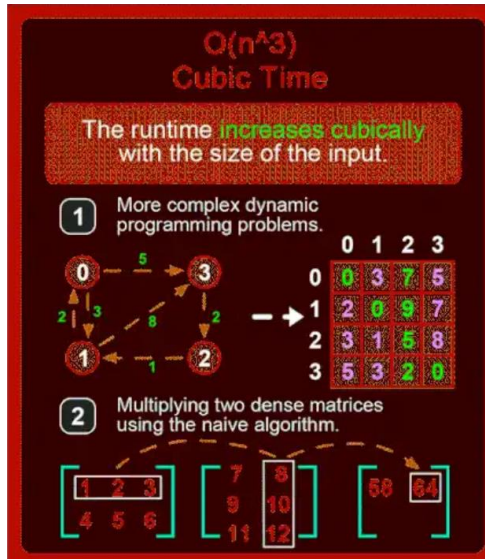
4. $O(n^2)$

Quadratic time notation. The runtime grows exponentially with input size. For example, simple sorting algorithms like bubble sort, insertion sort, and selection sort.



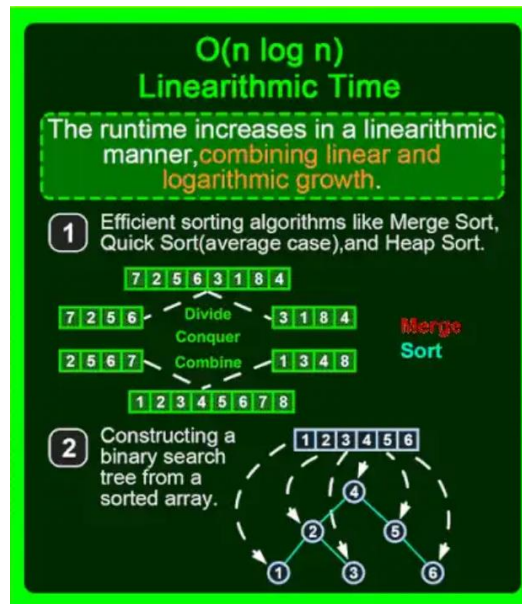
5. $O(n^3)$

Cubic time notation. The runtime escalates rapidly as the input size increases. For example, multiplying two dense matrices using the naive algorithm.



6. $O(n \log n)$

Linearithmic time notation. This is a blend of linear and logarithmic growth. For example, efficient sorting algorithms like merge sort, quick sort, and heap sort



7. $O(2^n)$

Exponential time notation. The runtime doubles with each new input element. For example, recursive algorithms solve problems by dividing them into multiple subproblems.

$O(2^n)$ Exponential Time

The runtime **doubles** with each additional element in the input.

1 Recursive algorithms that solve problems by dividing them into multiple subproblems.

The Traveling Salesman Problem

2 Solving the subset sum problem using recursion.

No alt text provided for this image

8. $O(n!)$

Factorial time notation. Runtime skyrockets with input size. For example, permutation-generation problems.

$O(n!)$ Factorial Time

The runtime **grows factorially** with the size of the input.

1 Permutation generation problems.

P

Q

R

S

Using 2 out of 4 boxes in a set

Possible arrangements: 12

PQ	QP	RP	SP
PR	QR	RQ	SQ
PS	QS	RS	SR

PERMUTATIONS

9. $O(\sqrt{n})$

Square root time notation. Runtime increases relative to the input's square root. For

example, searching within a range such as the Sieve of Eratosthenes for finding all primes up to n .

$O(\sqrt{n})$
Square Root Time

The runtime grows proportionally to the **square root** of the input size.

1 Algorithms that involve searching within a range, such as the Sieve of Eratosthenes for finding all primes up to n .

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100